DIGITAL IMAGE PROCESSING
July – November 2020

Chapter – 7

80386 Micro-Processor

Devang Patel Institute of Advance Technology and Research

FEATURES OF 80386:

- 32 bit processor, it has 32 bit ALU which allows to process 32 bit data at a time.
- 32 bit address bus, therefore it can access 4GB physical memory and 64Terabytes of Virtual memory.
- It has pipeline architecture which allows simultaneous instruction fetching, decoding, and executing and memory management.
- It allows user to switch between different OS such as DOS and UNIX
- Operates in Real, Protected and Virtual 8086 mode.

FEATURES OF 80386:

- It compatible with 8086,8088, 80186, 80286 architecture.
- It has Separate pins for its address and data line, this result in higher performance and easier hardware design.
- Prefetch unit permits to prefetch up to 16bytes of instruction code. Therefore fetch time for most instruction is hidden, increase the performance.

FEATURES OF 80386:

- Two versions of 80386 are commonly available:

80386DX	80386SX
32 bit address bus 32 bit data bus	24 bit address bus 16 bit data bus
Packaged in 132 pin ceramic pin grid array(PGA)	100 pin flat package
Address 4GB of memory	16 MB of memory

Signal Description of 80386

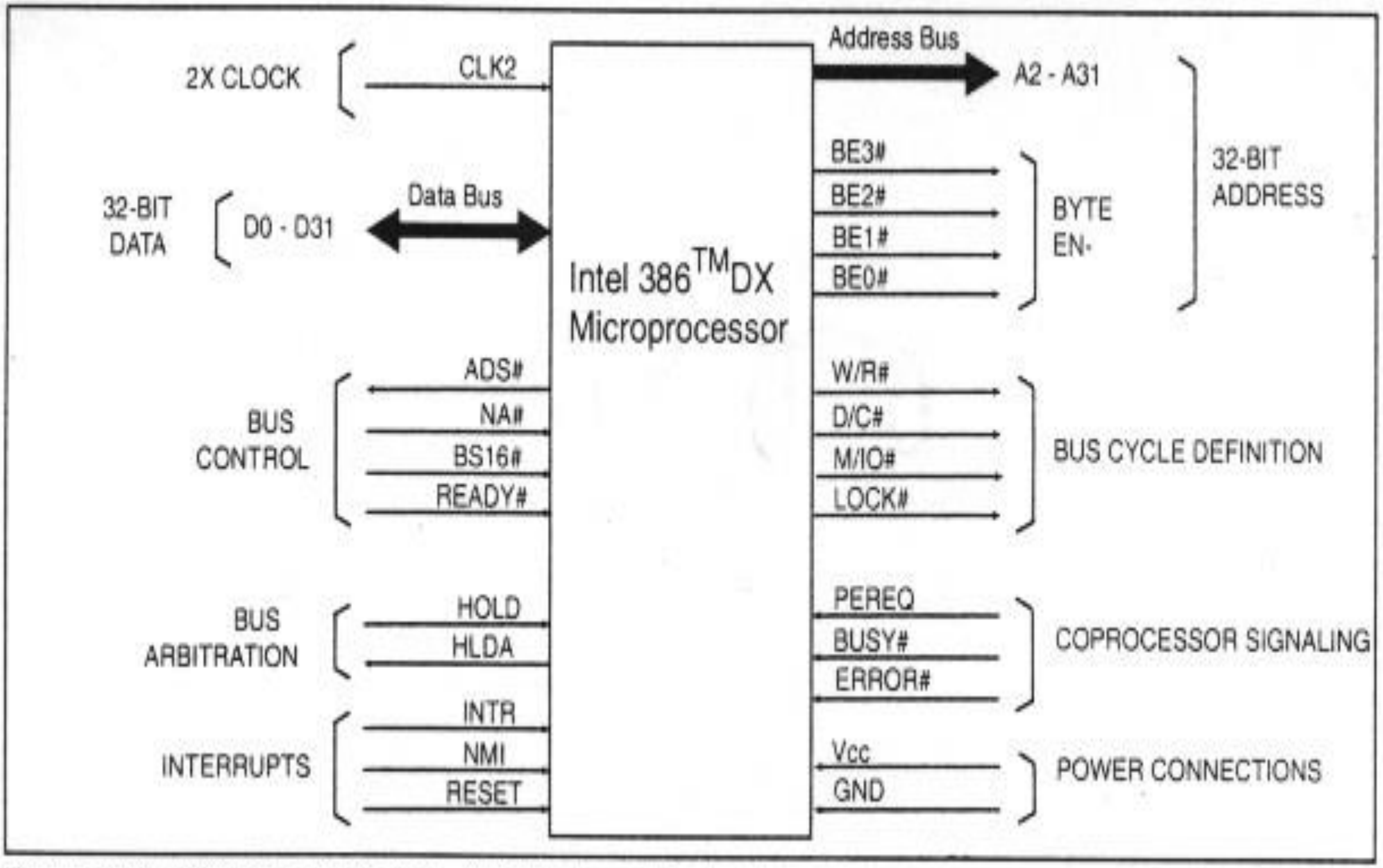


Figure 13-2. 80386 Block Diagram (# indicates active low)

(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1992)

Signal Descriptions of 80386

- **W/R#:**The write / read output distinguishes the write and read cycles from one another.
- **D/C#:**This data / control output pin distinguishes between a data transfer cycle from a machine control cycle like interrupt acknowledge.
- **M/IO#:**This output pin differentiates between the memory and I/O cycles.
- **LOCK#:**The LOCK# output pin enables the CPU to prevent the other bus masters from gaining the control of the system bus.

- **NA#:**The next address input pin, if activated, allows address pipelining, during 80386 bus cycles.
- **ADS#:**The address status output pin indicates that the address bus and bus cycle definition pins(W/R#, D/C#, M/IO#, BE0# to BE3#) are carrying the respective valid signals. The 80383 does not have any ALE signals and so this signals may be used for latching the address to external latches.
- **READY#:**The ready signals indicates to the CPU that the previous bus cycle has been terminated and the bus is ready for the next cycle. The signal is used to insert WAIT states in a bus cycle and is useful for interfacing of slow devices with CPU.
- **VCC:** These are system power supply lines.
- **VSS:** These return lines for the power supply

- **BS16#:**The bus size –16 input pin allows the interfacing of 16 bit devices with the 32 bit wide 80386 data bus. Successive 16 bit bus cycles may be executed to read a 32 bit data from a peripheral.
- **HOLD:** The hold input pin enables the other bus masters to gain control of the system bus if it is asserted.
- **HLDA:** The hold acknowledge output indicates that a valid bus hold request has been received and the bus has been relinquished by the CPU.
- **BUSY#:**The busy input signal indicates to the CPU that the coprocessor is busy with the allocated task.

- **ERROR#:**The error input pin indicates to the CPU that the coprocessor has encountered an error while executing its instruction.
- **PEREQ:** The processor extension request output signal indicates to the CPU to fetch a data word for the coprocessor.
- **INTR:** This interrupt pin is a maskable interrupt, that can be masked using the IF of the flag register.
- **NMI:**A valid request signal at the non-maskable interrupt request input pin internally generates a non-maskable interrupt.

- **RESET:** A high at this input pin suspends the current operation and restart the execution from the starting location.
- **N / C:** No connection pins are expected to be left open while connecting the 80386 in the circuit.

Architecture of 80386

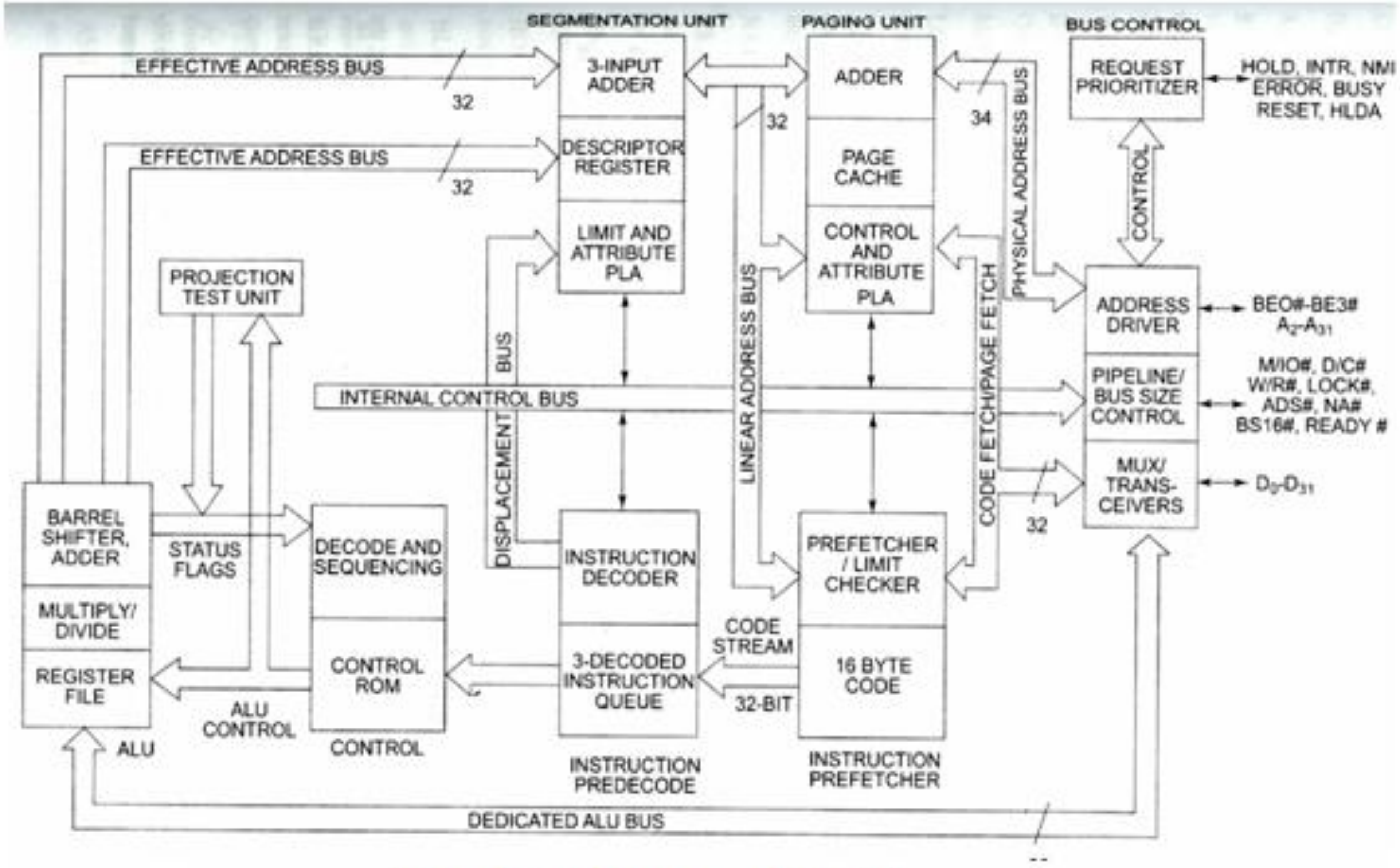


Fig: - Block Diagram of 80386

Architecture of 80386

- The Internal Architecture of 80386 is divided into 3 sections.
  - Central Processing Unit
    - Execution Unit
    - Instruction Unit
  - Memory management unit
    - Segmentation
    - Paging Unit
  - Bus interface unit

Architecture of 80386

1 Central processing unit(CPU)

a) Execution Unit : Reads the instruction from the instruction queue and execute the instruction.

Consists of three subunits : control, data and protection test unit

I) Control Unit: It contains microcode and special hardware allows processor to reduce time required for execution of multiply and divide instruction. It also speeds the effective address calculation.

II) Data Unit: Responsible for data operations requested by the control unit. It contains ALU, eight 32 bit general purpose registers and 64 bit barrel shifter. The barrel shifter is used for multiple bit shifts in one clock. Thus it increases the speed of all shift and rotate operations.

Architecture of 80386

III) Protection Test Unit: checks for segmentation violations under the control of the microcode.

b) Instruction Decode Unit : Takes instruction byte from the code prefetch queue and translates them in to microcode. The decoded instructions are then stored in the instruction queue.

2) Memory Management Unit

a) Segmentation Unit: Translate logical address to linear addresses at the request of execution unit. Compares the effective address for the length limit specified in the segment descriptor. Adds the segment base and the effective address to generate linear address. Before calculation of linear address it also checks access rights. It provides a 4 level protection mechanism for protecting and isolating the system code and data from those of the application program

Architecture of 80386

b) Paging Unit: Translate linear addresses generated by the segmentation unit into physical addresses. If paging unit is not enabled, the physical address is the same as the linear address. It give physical address to the Bus Interface Unit to perform memory and I/O accesses. It organizes the physical memory in terms of pages of 4Kbyts size each.

3) Bus Control Unit:

It provides a full 32 bit bi-directional data and 32- bit address bus. Responsible for following operations

i) Accepts internal requests for code fetch and for data transfers from the code fetch unit and from the execution unit. It then prioritize the request and generates signals to perform bus cycles.

Architecture of 80386

ii)It send address, data and control signals to communicate with memory and I/O devices

iii)It controls the interface to external bus masters and coprocessors.

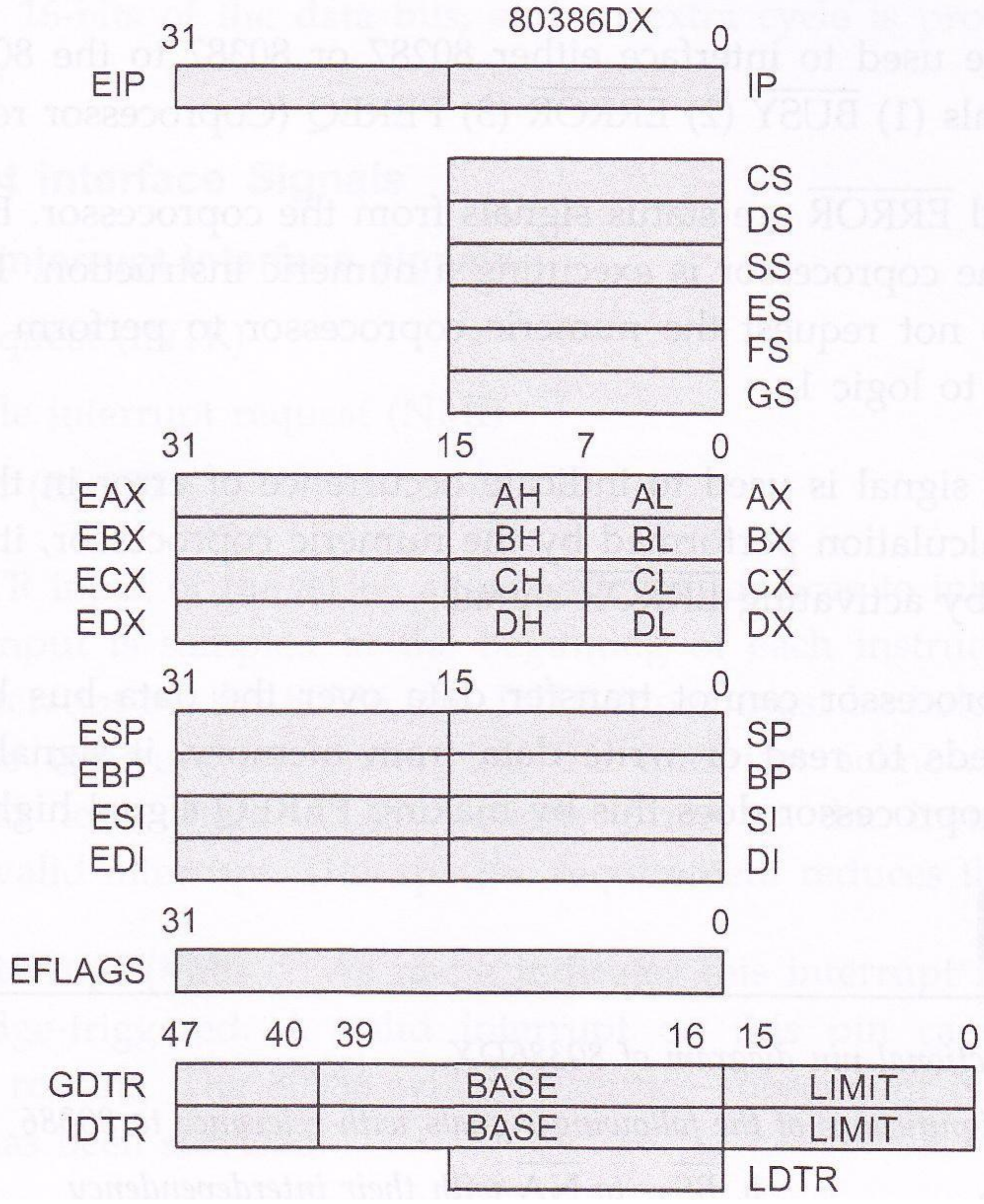
iv)It also provides the address relocation facility.

v) Instruction Prefetch Unit

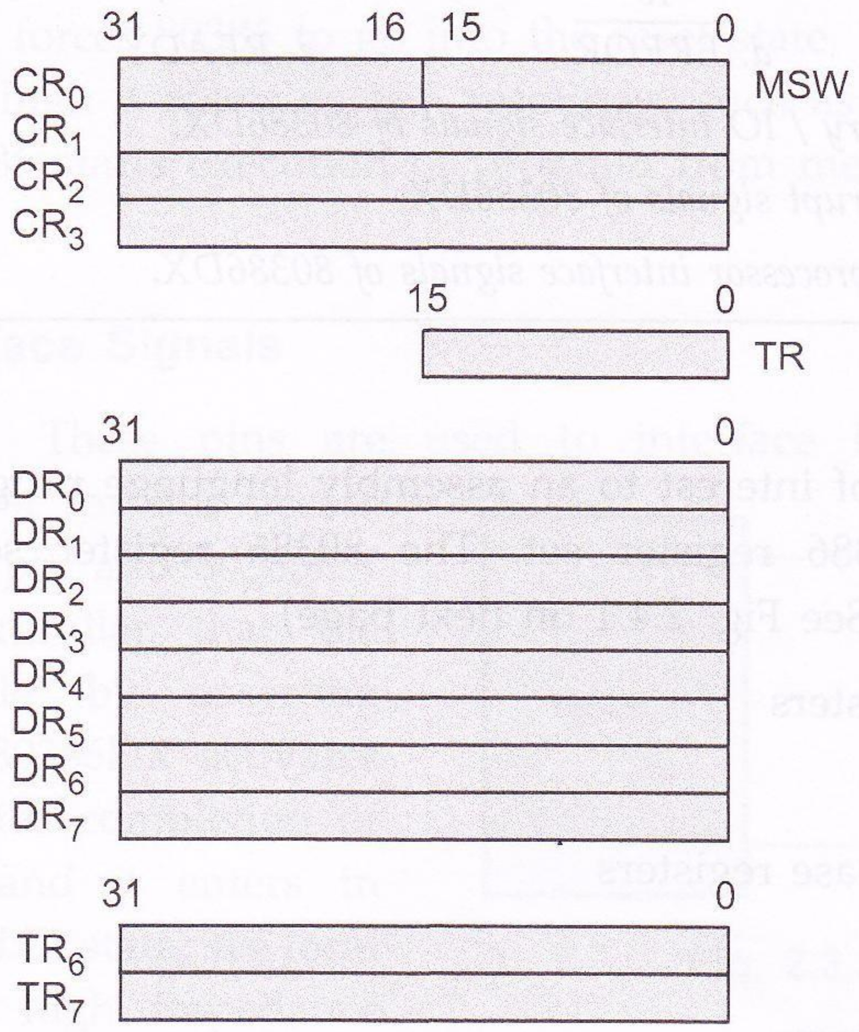
Fetches sequentially the instruction byte stream from the memory. It uses bus control unit to fetch instruction bytes when it is not performing bus cycles. These prefetched instruction bytes are stored in 16 bytes queue. When jump or call instructions are executed, the contents of the prefetched and decode queues are cleared out.

Register of 80386

- The 80386 has 32 registers in the following categories:
  - **General Purpose Register**
  - **Segment Registers**
  - **Instruction Pointer and Flags**
  - **Status and Control Registers**
  - **System Address Registers**
  - **Debug Registers**
  - **Test Registers**



80386DX
31
0
EIP
IP
CS
DS
SS
ES
FS
GS
31
15
7
0
EAX
AH
AL
AX
EBX
BH
BL
BX
ECX
CH
CL
CX
EDX
DH
DL
DX
31
15
0
ESP
SP
EBP
BP
ESI
SI
EDI
DI
31
0
EFLAGS
47
40
39
16
15
0
GDTR
BASE
LIMIT
IDTR
BASE
LIMIT
LDTR



31
16
15
0
CR_0
MSW
CR_1
CR_2
CR_3
15
0
TR
31
0
DR_0
DR_1
DR_2
DR_3
DR_4
DR_5
DR_6
DR_7
31
0
TR_6
TR_7

General Purpose Register

- The 80386 contains *32-bit general purpose register* called EAX, EBX, ECX, EDX, ESP, EBP, ESI, and EDI.

- The *lower 16-bits of each* of the general purpose register can be accessed individually.

- These 16-bit registers are accessed as AX, BX, CX, DX, SP, BP, SI, and, DI respectively.

- The AX, BX, CX and DX registers can be further divided into Two separate bytes : Higher byte and lower byte.

Segment Register

- The 80386 has a *1 MB address space* in *real mode.*
- But all of this memory *cannot be active at one time*.
- It supports *six simultaneously accessible memory blocks* called segments.
- A segment represents an independently accessible block of memory consisting of 64 K consecutive byte-wide storage locations.
- These segments are addressed by 16-bit registers : CS, DS, ES, SS, FS and GS.

Segment Register

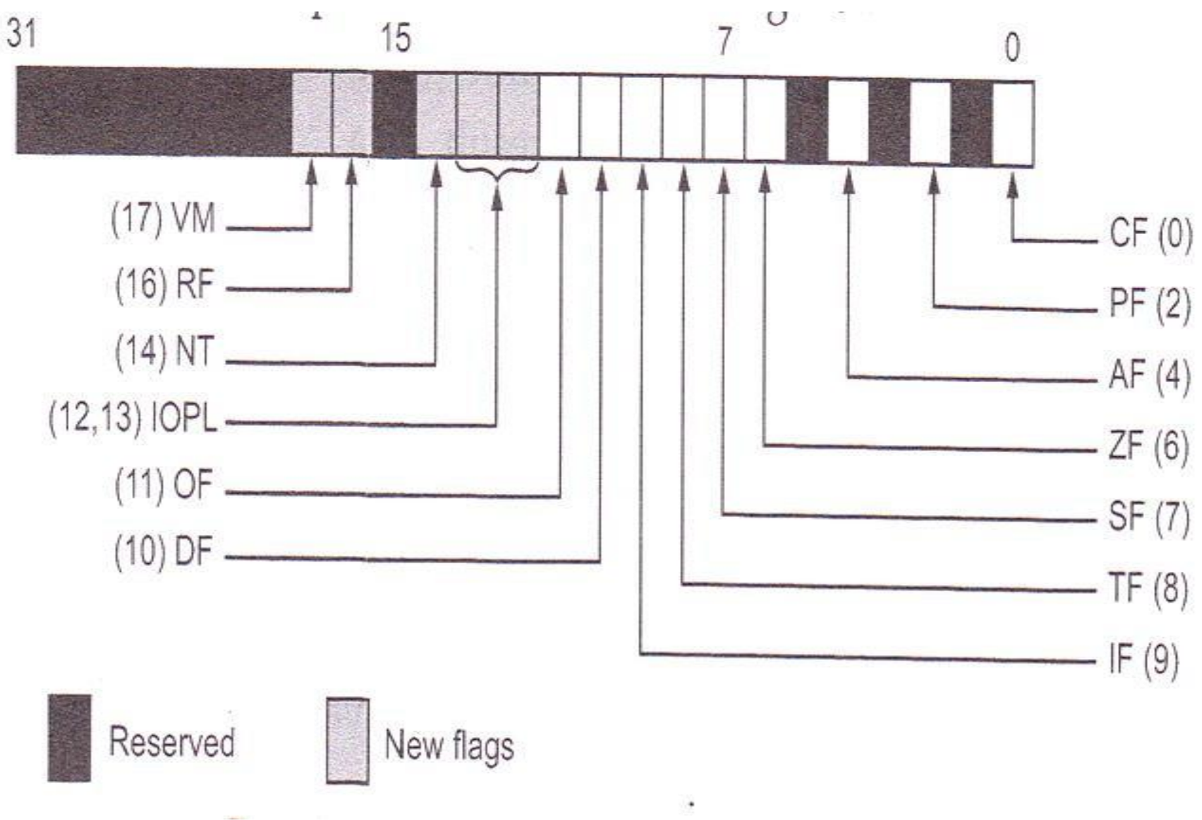
1. CS (Code Segment) : holds the base address of the currently active code segment.

2. DS (Data Segment) : is used to hold the base address of currently active data segment.

3. ES, FS, & GS (Extra Segment) : are used as general data segment registers.
   - These registers hold the base addresses of three different memory segments.
   - These segments are referred as to Extra Segments.

4. SS (Stack Segment) : The base address of the currently Active stack segment.

Flag Register



- Six Conditional Flags
  - Carry Flag (CF)
  - Parity Flag (PF)
  - Auxiliary Flag(AF)
  - Zero Flag (ZF)
  - Sign Flag (SF)
  - Overflow Flag (OF)
- Three Control Flags
  - Interrupt Flag (IF)
  - Trap Flag (TF)
  - Direction Flag (DF)
- Four System Flags
  - Input/output privilege level (IOPL)
  - Nested Task (NT)
  - Resume Flag (RF)
  - Virtual Mode Flag (VM)

Flag Register

1. VM (Virtual Memory) flag :
   - *Indicates operating mode of 80386.*
   - When VM flag is set, 80386 *switches from protected mode to virtual 8086 mode.*
2. R (Resume) flag :
   - This flag, when set *allows selective masking* of some *exceptions* at the time of debugging.
3. NT (Nested flag) :
   - This flag is set when one *system task invokes another* task. (i.e. nested task).
4. IOPL (I/O Privilege level) :
   - The *two bits* in the IOPL are used by the processor and the operating system to *determine* your *application's access to I/O facilities.*

Flag Register

5. IF (Interrupt Flag) :
    - When interrupt flag is set, the 80386 *recognizes and handles external* hardware interrupts on its INTR pin.
    - If the interrupt flag is cleared, 80386 ignores any inputs on this pin.
    - The IF flag is set and cleared with the STI and CLI instructions, respectively.

6. TF (Trap Flag) :
    - Trap flag allows user to *single-step through programs*.
    - An 80386 detects that this flag is set, it executes one instruction and then automatically generates an internal exception 1.
    - After servicing the exception, the processor executes next instruction and repeats the process.
    - This *single stepping continues until prgram code resets this flag* for debugging programs single step facility is used.

System Address Registers

System Adderss Register

There are four system address register :

1. TR (Task Register)
2. IDTR (Interrupt Descriptor Table Register)
3. GDTR (Global Descriptor Table Register)
4. LDTR (Local Descriptor table Register).

- These registers *hold the addresses for the four special descriptor table segments.*

  - Global descriptor table (GDT),
  - Local descriptor table (LDT),
  - Interrupt descriptor table (IDT), and
  - Task state segment descriptor (TSS).

System Adderss Register

1. TR (Task Register)
   - Points to the *Task state segment*

2. IDTR (Interrupt Descriptor Table Register)
   - points to the *Interrupt Descriptor table (IDT)*

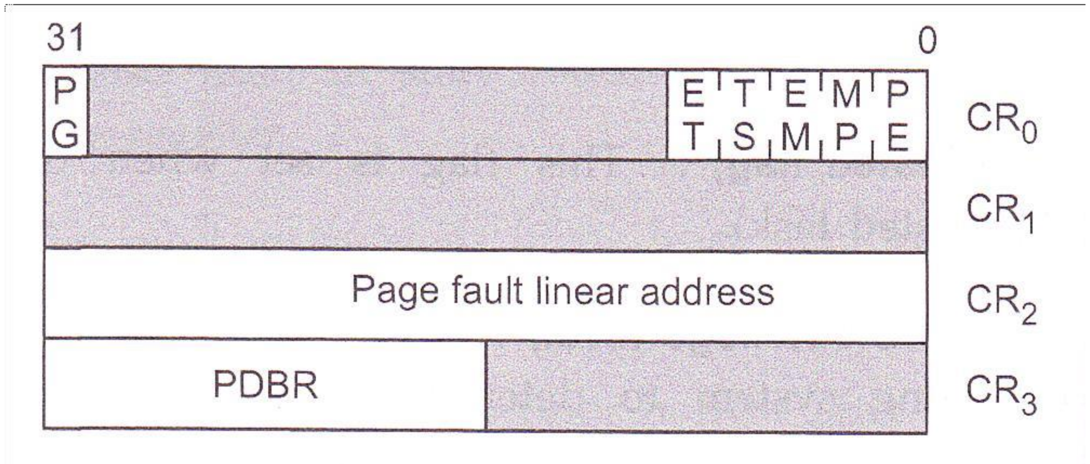
3. GDTR (Global Descriptor Table Register)
   - points to the *Global Descriptor Table (GDT)*

4. LDTR (Local Descriptor Table Register)
   - points to the *Local Descriptor Table (LDT)*

Control Register

Control Register

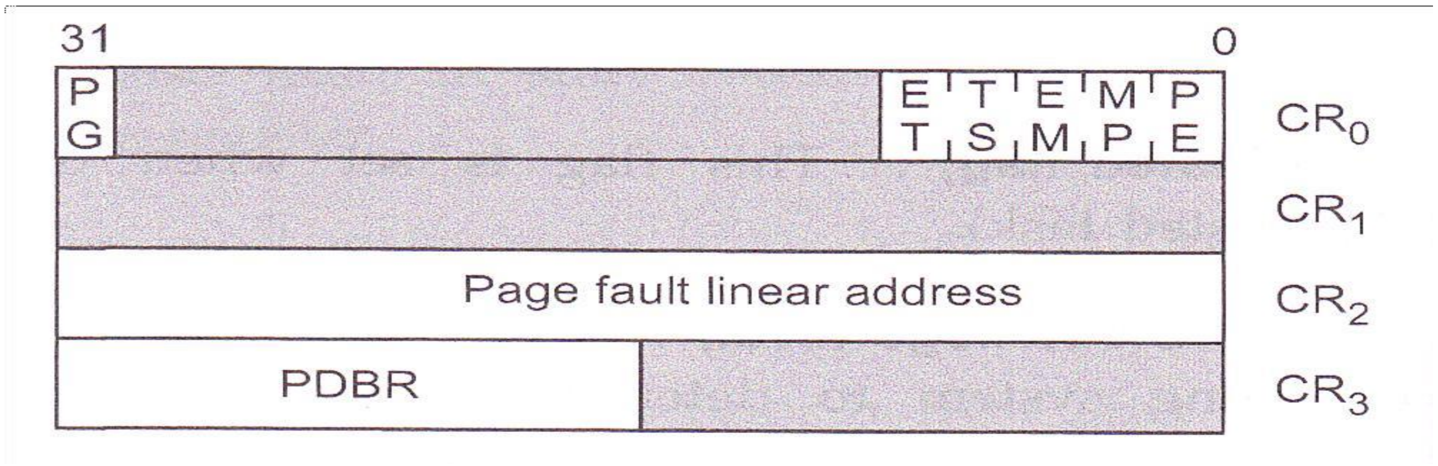
- There are *four* control registers :
- CR0, CR1, CR2 and CR3.
- These registers *define the machine state* that *affects all the tasks* in the systems.



Control Register

CR0 : Holds the MSW (Machine Status Word).

- It contains six status bits :
  1. PE (Protection Enable)
  2. MP (Math Present)
  3. EM (Emulate Coprocessor)
  4. TS (Task Switched)
  5. ET (Extension Type)
  6. PG (Paging).



Control Register

1. PE (Protection Enable) :

    - This bit is *similar to the VM* bit in EFLAGs in that it controls the 80386's mode of operation.
    - When PE is set, it is in *Protection mode* otherwise it *operates in Real Mode.*

2. MP (Math Present) :

    - When this bit is set, the *80386 assumes that real floating point hardware (80287 or 80387) is present* in the system.
    - When this bit is clear, the 80386 assumes that no such coprocessor exists, and will not attempt to use real floating point hardware.

Control Register

3. EM (Emulate coprocessor) :
   - When this bit is set, the *80386 will generate an exception 11 (device not available) whenever it attempts to execute a floating point instruction.*
   - Programmer can use this exception handler to emulate floating point hardware in software.

4. TS (Task Switched) :
   - The 80386 *sets the bit automatically every time it performs a task switch.*
   - *It will never clear this bit on its own*. But *programmer can clear* this bit using *CLTS instruction*.

Control Register

5. ET (Extension Type) :
   - When power is applied, *80386 detects whether numeric Processor connected is 80287 or 80387 & sets ET to logic 1 , if numeric processor is 80387.*
   - This is necessary because the 80387 uses a slightly different protocol than 80287.

6. PG (Paging) :
   - *This bit enables or disables paging mechanism* in Memory Management Unit (MMU).
   - If bit is set, paging is enabled.

Control Register

1. ### Control Register 1 (CR1)
   - This is *reserved by Intel.*

2. ### Control Register 2 (CR2)
   - CR2 is read-only register.
   - The *80386, itself writes the last 32-bit linear address* of page fault routine in this register.
   - When page fault occurs, the 80386 generates exception 14 (page fault)
   - This address *is important for writing page fault routine*.
   - The page *fault routine helps programmer to find cause of the fault.*

Control Register

3. Control Register 3 (CR3)

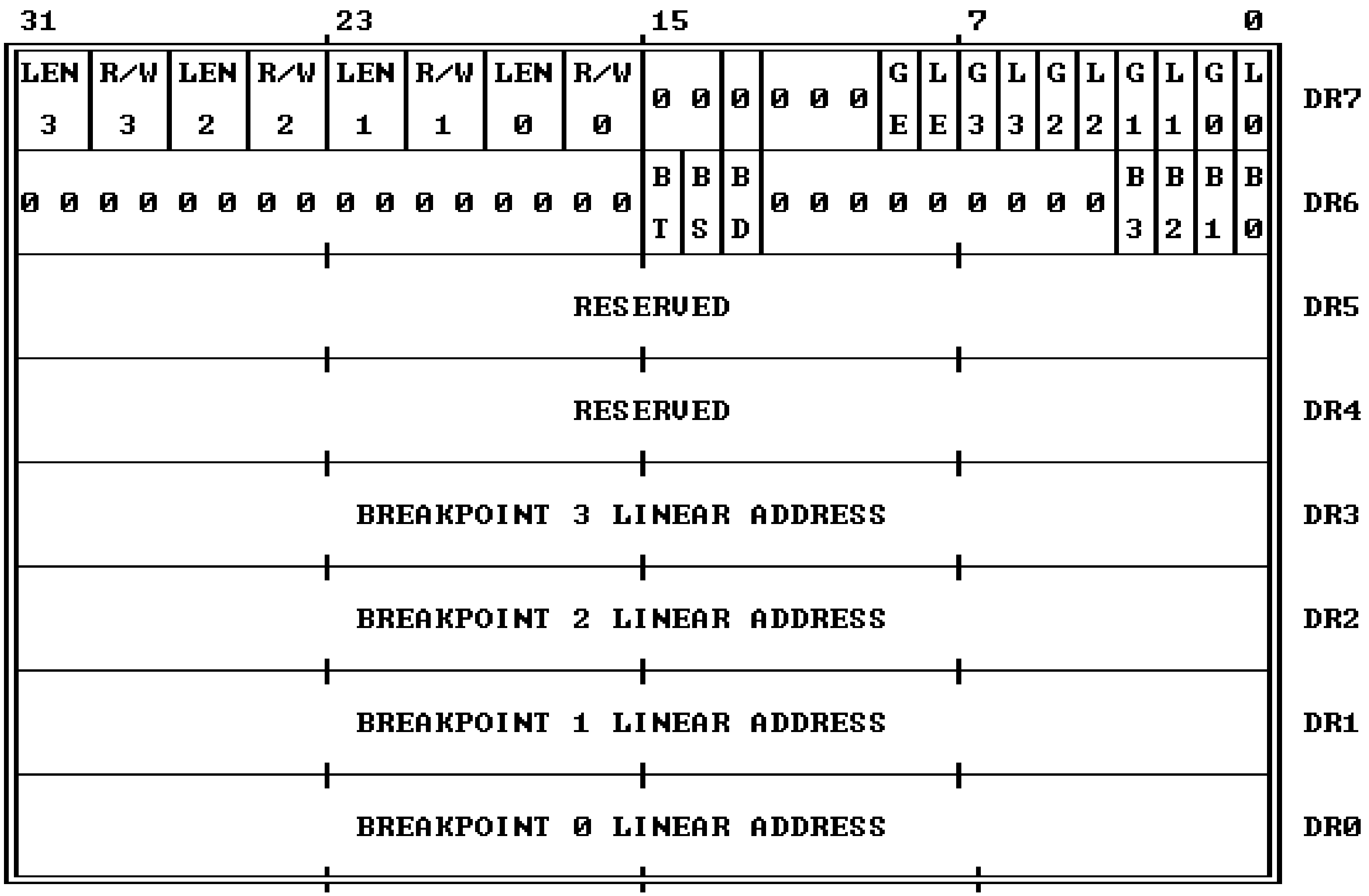
   - Control register 3 *holds the physical address of the root of the two level paging tables,* used when paging is enabled. It is also called *Page Directory Base Register*

Debug Register

- Intel has provide a set of 8 debug registers for hardware debugging. Out of these eight registers DR0 to DR7, two registers DR4 and DR5 are Intel reserved.
- The initial four registers DR0 to DR3 store four program controllable breakpoint addresses, while DR6 and DR7 respectively hold breakpoint status and breakpoint control information.
- Breakpoint address may locate an instruction or data.

Debug Register

Figure 12-1. Debug Registers



NOTE
0 MEANS INTEL RESERVED. DO NOT DEFINE.

Debug Address Register (DR0-DR3)

- Each of these registers contains the linear address associated with one of four breakpoint conditions. Each breakpoint condition is further defined by bits in DR7.

- The debug address registers are effective whether or not paging is enabled. The addresses in these registers are linear addresses.

Debug Status Register(DR7)

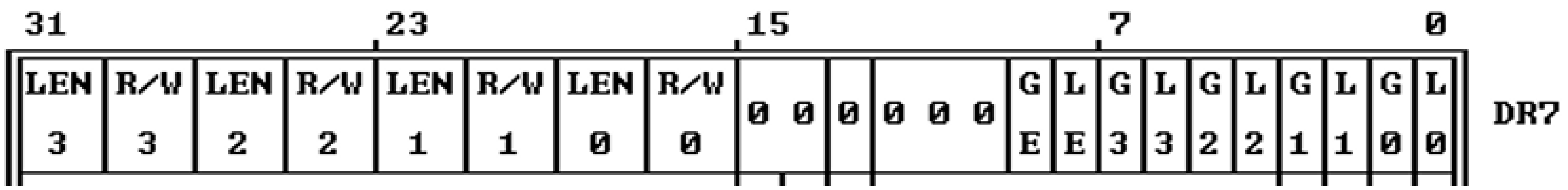
- ❖ The debug status register permits the debugger to determine which debug conditions have occurred.
- ❖ For each address in registers DR0-DR3, the corresponding fields R/W0 through R/W3 specify the type of action that should cause a breakpoint. The processor interprets these bits as follows:

  00 -- Break on instruction execution only

  01 -- Break on data writes only

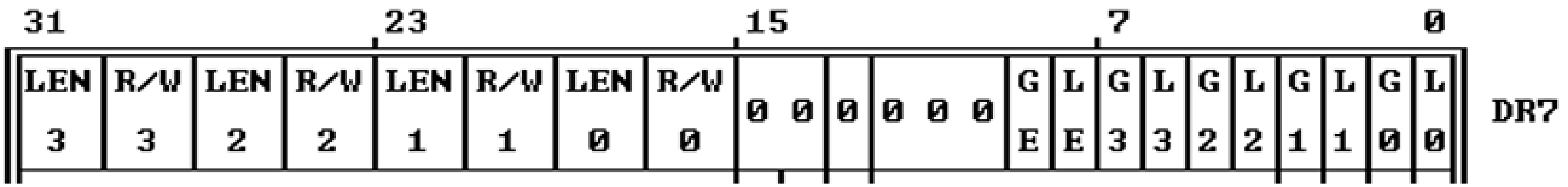
  10 -- undefined

  11 -- Break on data reads or writes but not instruction fetches



Debug Status Register(DR7)

- ❖ The LE and GE bits control the "exact data breakpoint match" feature of the processor.
- ❖ If either LE or GE is set, the processor slows execution so that data breakpoints are reported on the instruction that causes them.
- ❖ It is recommended that one of these bits be set whenever data breakpoints are armed.
- ❖ The processor clears LE at a task switch but does not clear GE.



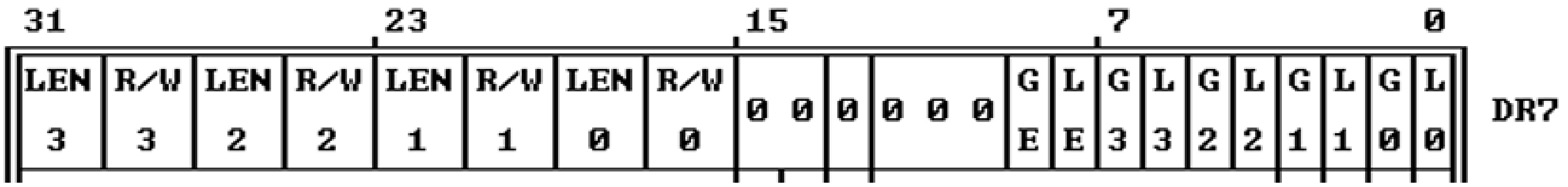
Debug Status Register(DR7)

•Fields LEN0 through LEN3 specify the length of data item to be monitored.

•A length of 1, 2, or 4 bytes may be specified. The values of the length fields are interpreted as follows:

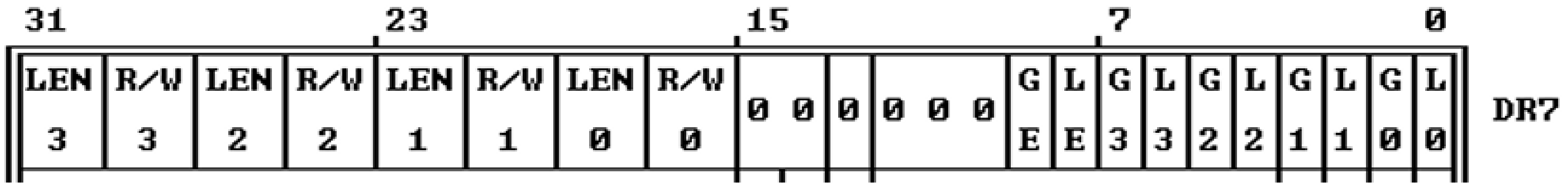
00 -- one-byte length
01 -- two-byte length
10 -- undefined
11 -- four-byte length

• If RWn is 00 (instruction execution), then LENn should also be 00. Any other length is undefined.



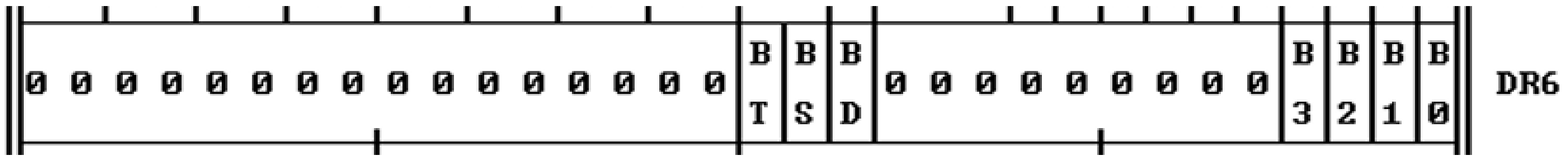
Debug Status Register(DR7)

- The low-order eight bits of DR7 (L0 through L3 and G0 through G3) selectively enable the four address breakpoint conditions.
- There are two levels of enabling: the local (L0 through L3) and global (G0 through G3) levels.
- The local enable bits are automatically reset by the processor at every task switch to avoid unwanted breakpoint conditions in the new task.
- The global enable bits are not reset by a task switch; therefore, they can be used for conditions that are global to all tasks.



Debug Status Register(DR6)

- When the processor detects an enabled debug exception, it sets the low-order bits of this register (B0 thru B3) before entering the debug exception handler.
- Bn is set if the condition described by DRn, LENn, and R/Wn occurs.
- The BT bit is associated with the T-bit (debug trap bit) of the TSS.



Debug Status Register(DR6)

- The processor sets the BT bit before entering the debug handler if a task switch has occurred and the T-bit of the new TSS is set.
- There is no corresponding bit in DR7 that enables and disables this trap; the T-bit of the TSS is the sole enabling bit.
- The BS bit is associated with the TF (trap flag) bit of the EFLAGS register.

Debug Status Register(DR6)

- The BS bit is set if the debug handler is entered due to the occurrence of a single-step exception
- The BD bit is set if the next instruction will read or write one of the eight debug registers and ICE-386 is also using the debug registers at the same time.
- **The bits of DR6 are never cleared by the processor.**

Physical Address Space

Physical Address Space

- The physical memory of an 80386 system is organized as a sequence of 8-bits.
- Each byte is assigned a unique address that ranges from zero to a maximum of $2^{(32)} – 1$(4 gigabytes).
- 80386 programs/ however, are independent of the physical address space.

Physical Address Space

- This means that programs can be written without knowledge of how much physical memory is available and without knowledge of exactly where in physical memory the instructions and data are located.

- The model of memory organization seen by applications programmers is determined by systems-software designers.

- The architecture of the 80386 gives designers the freedom to choose a model for each task.

Physical Address Space

The model of memory organization can range between the following extremes:

1. Flat address space
2. Segmented address space

1. Flat address space :

- In a "flat" model of memory organization, the applications programmer sees a single array of up to $2^{(32)}$ bytes (4 gigabytes).
- The processor maps the 4 gigabyte flat space onto the physical address space by the address translation mechanisms.

Physical Address Space

2. **Segmented address space :**

- A segmented model consisting of a collection of up to 16,383 linear address spaces of up to 4 gigabytes each.
- In a segmented model of memory organization, the address space as viewed by an applications program (called the logical address space) is a much larger space of up to $2^{(46)}$ byte (64 terabytes).
- The processor maps the 64 terabyte logical address space onto the physical address space (up to 4 gigabytes) by the address translation mechanisms.
- Both models can provide memory protection. Different tasks may employ different models of memory organization.

ADDRESSING MODES

- The 80386 supports overall eleven addressing modes to facilitate efficient execution of higher level language programs.
- In case of all those modes, the 80386 can now have 32-bit immediate or 32- bit register operands or displacements.
- The 80386 has a family of scaled modes.
- In case of scaled modes, any of the index register values can be multiplied by a valid scale factor to obtain the displacement.
- The valid scale factor are 1, 2, 4 and 8.

ADDRESSING MODES

- *Register addressing Mode*
- *Immediate addressing Mode*
- *Direct addressing Mode*
- *Register Indirect addressing Mode*
- *Based addressing Mode*
- *Index addressing Mode*
- *Scaled Index addressing Mode*
- *Based Index addressing Mode*
- *Based Scaled Index addressing Mode*
- *Based Index addressing Mode with Displacement*
- *Based Scaled Index addressing Mode with Displacement*

ADDRESSING MODES

- Scaled Indexed Mode:
    - Contents of the an index register are multiplied by a scale factor that may be added further to get the operand offset.
- Based Scaled Indexed Mode:
    - Contents of the an index register are multiplied by a scale factor and then added to base register to obtain the offset.

ADDRESSING MODES

- Based Scaled Indexed Mode with Displacement:
  - The Contents of the an index register are multiplied by a scaling factor and the result is added to a base register and a displacement to get the offset of an operand.

Real Addressing Modes of 80386

- When processor Reset or Powered up then 80386 is initialized in REAL Mode.
- Real mode has same base architecture as that of 8086, but allows access to the 32-bit register set of 80386.
- The addressing mechanism, memory size, interrupt handling , are identical to Real Mode of the 80286.
- All of the 80386 instructions are available in Real Mode is 16-bits, same as that of 8086.
- To use 32-bit registers and addressing modes, override prefixes ‘E’ must be used.
- The segment size on the 80386 in Real mode is 64KB so 32-bit effective addresses must be less than 0000FFFFH.
- The basic Purpose of Real mode in 80386 is to set up the processor for Protected Mode Operation.

Memory Addressing in Real Mode of 80386

- In Real mode the max. memory size is limited to 1MB, so only address line A0-A19 are used.
- In Real addressing mode the linear addresses are the same physical addresses as paging is not allowed.
- Physical address is calculated in Real mode by adding the content of segment register after shifting it left by four bits with an effective address and generates a physical address. Which is compatible with 80286 Real Mode.

- All address in Real Mode should not more than 64KB in size and may be Read, written or executed.
- The 80386 generates an exception 13 if a data operand or instruction fetch occurs past the end of a segment.
- Segment may be overlapped in Real mode means if a particular segment does not use all 64KB another segment can be override on top of the unused portion of the previous segment.
- Hence, this allows the programmer to minimize the amount of physical memory needed for a program.

Protected Virtual Address Mode (PVAM) of 80386

- When the processor operates in protected virtual address mode i.e. Protected Mode then the complete capabilities of the 80386 are used.
- Protected mode is used to increase the linear address space up to 4GB and used to execute virtual memory programs of the size of 64TB.
- Also Protected mode of 80386 is used to run all the existing 8086 and 80286 program with a memory management and protection mechanism.
- Protected mode of 80386 also supports the use of additional instructions for supporting multitasking OS.

- The linear address is then either used as the 32- bit physical address, or the paging mechanism maps the 32- bit linear address into a 32 bit physical address if paging is enabled.
- In protected mode the selector is used as an index for system define table such as LDT or GDT
- The LDT or GDT table contains 32 bit base address of given segment.
- The physical address is formed by adding the base address obtained from the table to offset.
- Paging provides additional memory management mechanism which operates only in Protected mode and managing the very large segments of the 80386.

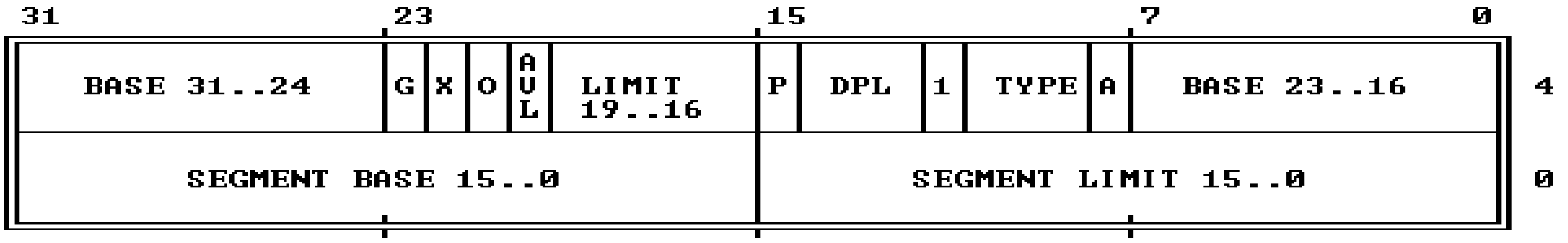
Segment Descriptor Cache

- Every segment register has a segment descriptor cache register.
- **Selector fields:** In protected mode of 80386, a selector has three fields i.e. Table indicator (TI) which is used to points either a Local or Global descriptor table, descriptor entry index (Index) and Requestor Privilege Level (RPL)
- The TI bit is used to select one of two tables of descriptors. i.e. GDT if TI bit is 0
- And TI is 1 then LDT

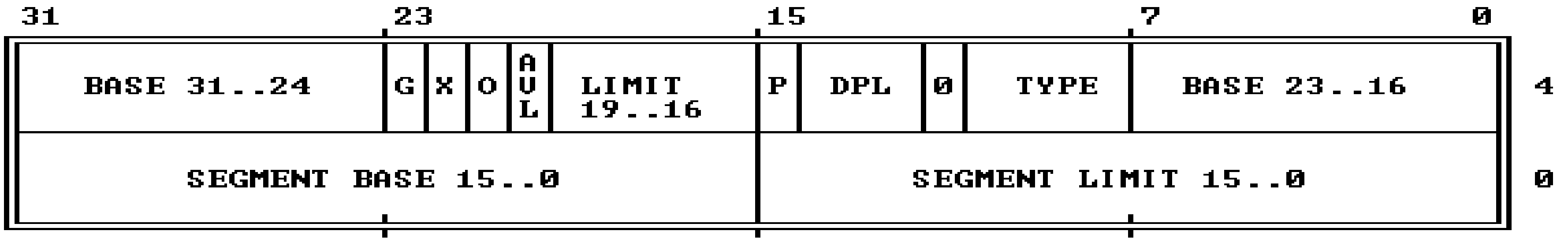
- The Index is used to selects one of 8KB descriptors entry in appropriate descriptor table.
- The RPL (Requestor Privilege Level) bits are used for testing of the selector’s privilege attributes.

Descriptor Attributes Bits

DESCRIPTORS USED FOR APPLICATIONS CODE AND DATA SEGMENTS



DESCRIPTORS USED FOR SPECIAL SYSTEM SEGMENTS



- The segment descriptor provides the processor with the data it needs to map a logical address into a linear address.
- Descriptors are created by compilers, linkers, loaders, or the operating system, not by applications programmers.

Descriptor Attributes Bits

- The A (accessed) attributed bit indicates whether the segment has been accessed by the CPU or not.
- The TYPE field decides the descriptor type.
- The S bit decides whether it is a system descriptor (S=0) or code/data segment descriptor (S=1).
- The DPL field specifies the descriptor privilege level.

Descriptor Attributes Bits

- The D bit specifies the code segment operation size. If D=1, the segment is a 32-bit operand segment, else, it is a 16-bit operand segment.
- The P bit (present) signifies whether the segment is present in the physical memory or not. If P=1, the segment is present in the physical memory.
- The G (granularity) bit Specifies the units with which the LIMIT field is interpreted. When the bit is clear, the limit is interpreted in units of one byte; when set, the limit is interpreted in units of 4 Kilobytes.
- The AVL (available) field specifies whether the descriptor is for user or for operating system.

Difference between Real and Protected Mode

Real Mode	Protected Mode (PVAM)
Memory addressing up to 1 MB physical memory	Memory addressing up to 16 MB of physical memory
No virtual memory support	Supports up to 64TB of virtual memory
Memory Protection mechanism is not available	Memory Protection Mechanism is available
Does not support virtual address space	Gives virtual and physical address space
Does not support LDT and GDT	Supports LDT and GDT
Segment descriptor cache is not available	Segment descriptor cache is available
Supports Segmentation	Supports segmentation and paging.

Segmentation

- Segmentation is memory management technique with protection.
- The information of the segment is stored in an 8 byte format called as descriptor.
- There are three types of descriptor table:
  - LDT
  - GDT
  - IDT

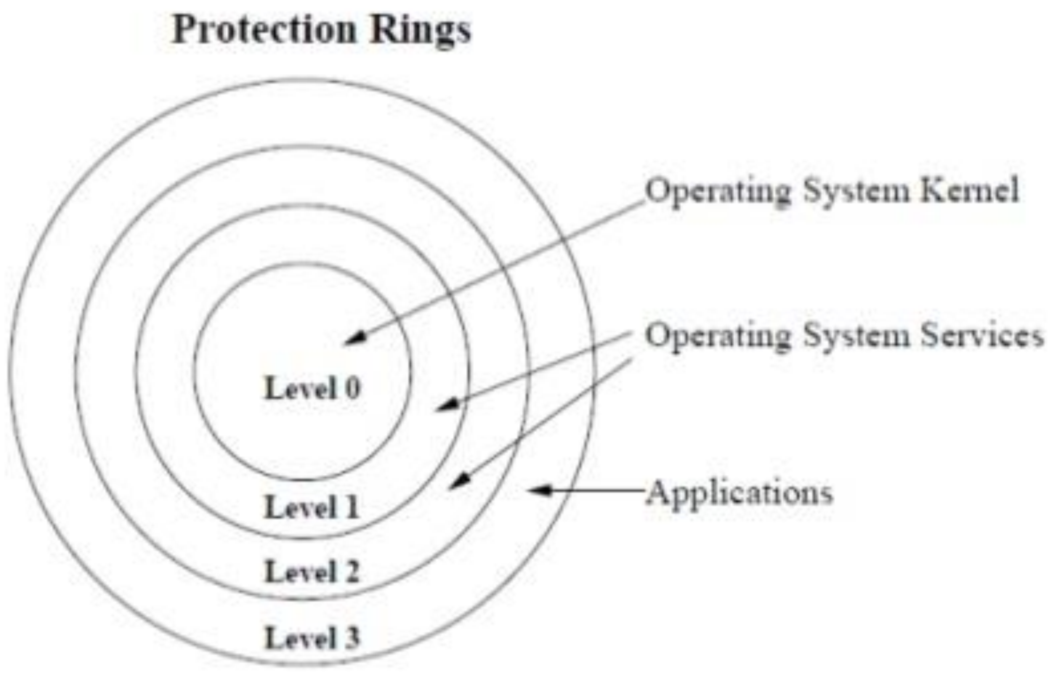
- Each table content maximum 8K of 8byte descriptors and the upper 13 bits of selectors are used as an index for the descriptor table.
- These tables have registers associated with them which hold the 32-bit linear base address and the 16- bit limit of each table and these registers are LDTR, GDTR and IDTR.

Protection Levels in 80386

- The 80386 supports four protection which isolates and protect user programs for each other and the operating system during a multi-tasking operating system.
- The protection level avoid the use of privileged instructions, I/O instructions and access to segments and segment descriptors.
- The privilege protection mechanism of 80386 is integrated in On- chip Memory Management Unit which also gives protection to pages, when paging is enable.

Privilege protection

- 80386 protection mechanism
  - Memory management
  - Privilege protection
- 4 privilege level protection
  - PL0 (highest)
  - PL1
  - PL2
  - PL3(lowest)
- A numerically
Smaller PL means a
Higher privilege.



Privilege levels in 80386

A. **Task Privilege level:**
   - At any point in time, a task in 80386 always executes at one of four Privilege levels.
   - The current privilege level of selector is indicated by the RPL field. i.e. the two least significant bits of the selector.

B. **Selector Privilege (RPL):**
   - The privilege level of selector is indicated by the RPL field i.e. two least significant bits of the selector.
   - RPL of selector is used only to test requestor privilege level with current privilege level of a segment to be used called effective privilege level i.e. EPL

Virtual Mode 8086

- In the 80386 microprocessor and later, virtual 8086 mode (also called virtual real mode, V86-mode or VM86) allows the execution of real mode applications that are incapable of running directly in protected mode while the processor is running a protected mode operating system.
- It is a hardware virtualization technique that allowed multiple 8086 processors to be emulated by the 386 chip; it emerged from the painful experiences with the 80286 protected mode, which by itself was not suitable to run concurrent real mode applications well.
- VM86 mode uses a segmentation scheme identical to that of real mode (for compatibility reasons) which creates 20-bit linear addresses in the same manner as 20-bit physical addresses are created in real mode, but are subject to protected mode's memory paging mechanism.

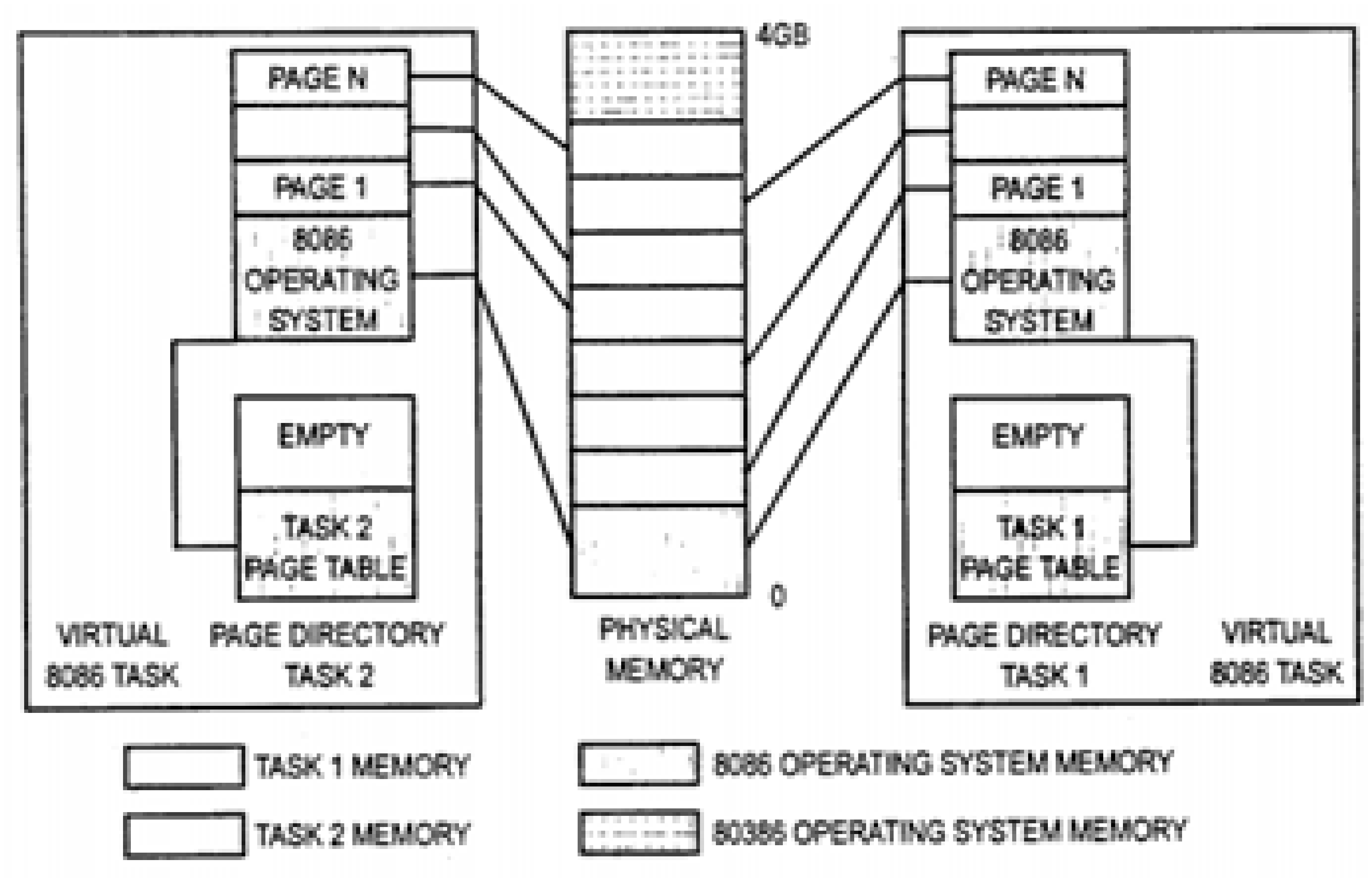
Virtual Mode 8086

- To use virtual 8086 mode, an operating system sets up a virtual 8086 mode monitor, which is a program that manages the real-mode program and emulates or filters access to system hardware and software resources.
- The monitor must run at privilege level 0 and in protected mode.
- Only the 8086 program runs in VM86 mode and at privilege level 3.
- When the real-mode program attempts to do things like access certain I/O ports to use hardware devices or access certain regions in its memory space, the CPU traps these events and calls the V86 monitor, which examines what the real mode program is trying to do and either acts as a proxy to interface with the hardware, emulates the intended function the real-mode program was trying to access, or terminates the real-mode program if it is trying to do something that cannot either be allowed or be adequately supported (such as reboot the machine, set a video display into a mode that is not supported by the hardware and is not emulated, or write over operating system code).

Virtual Mode 8086

- The V86 monitor can also deny permission gently by emulating the failure of a requested
- Also, the V86 monitor can do things like map memory pages, intercept calls and interrupts, and preempt the real-mode program, allowing real-mode programs to be multitasked like protected-mode programs.
- By intercepting the hardware and software I/O of the real-mode program and tracking the state that the V86 program expects, it can allow multiple programs to share the same hardware without interfering with each other.
- So V86 mode provides a way for real-mode programs designed for a single-tasking environment (like DOS) to run concurrently in a multitasking environment.

Virtual 8086 mode for executing 80386 Programs



- The 80386 allows the execution of 8086 program in both Real Mode and Virtual 8086 mode
- Virtual 8086 mode provides the system designer the most flexibility out of two modes i.e Real and PVM.
- The virtual mode allows the execution 8086 programs, by taking full advantage of the 80386 protection mechanism.
- Hence, it allows the simultaneous execution of 8086 OS with its application, and 80386 OS, both 8086 and 80386 applications.

Virtual 8086 Mode Addressing Mechanism

- The major difference between 80386 Real and Protected mode is the way that segment selectors are interpreted.
- When the processor is operating in Virtual Mode the segment registers are used in an identical to Real Mode.
- The content of the segment register are shifted towards left by 4 bits and added to the offset to form the segment base linear address.
- The 80386 gives facility to the OS for specifying which programs use 8086 style address mechanism, and which program use Protected mode addressing on a per task basis.

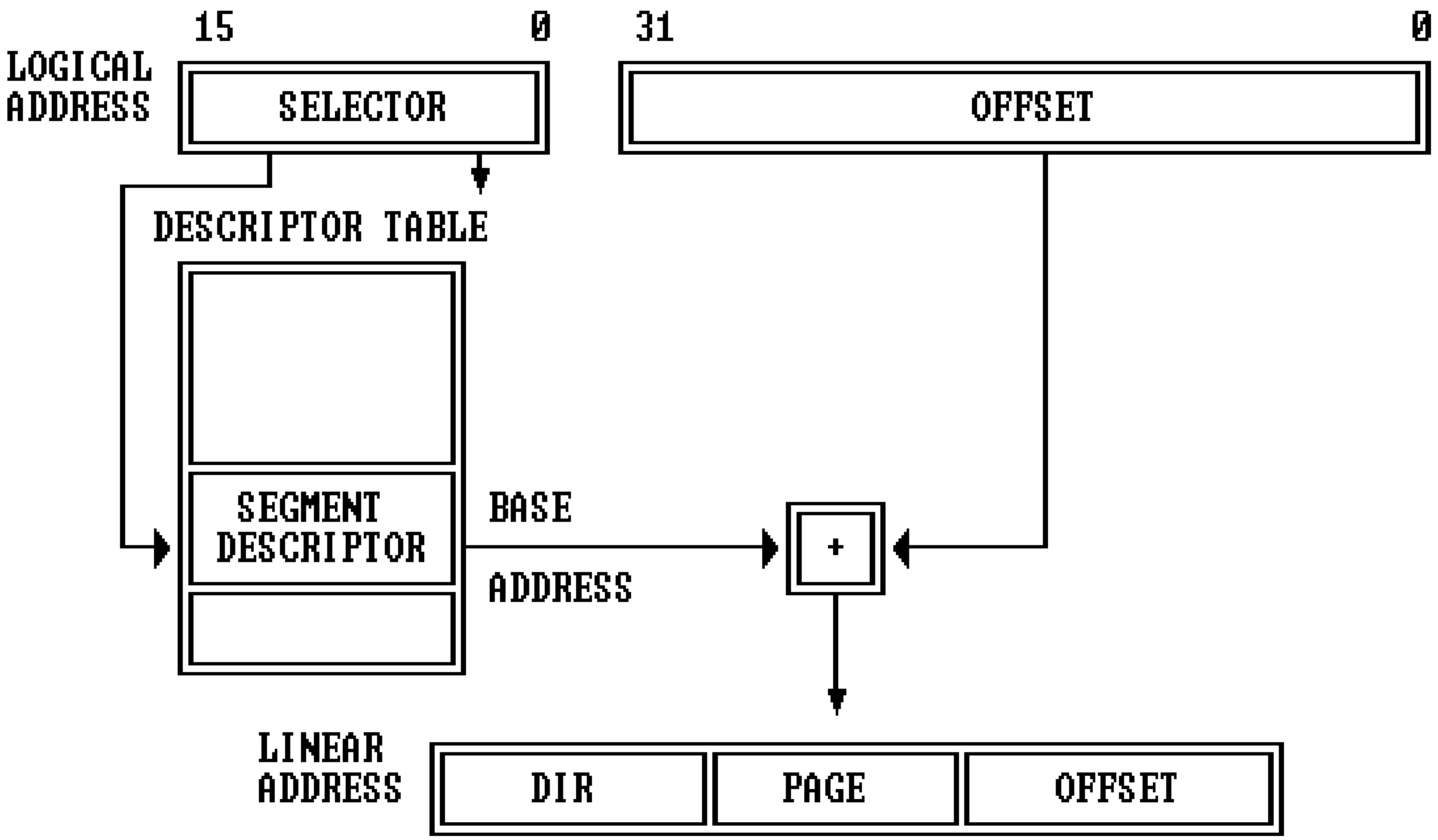
- By the use of paging memory management, the one MB address space of virtual mode task can be mapped anywhere in the 4 GB of linear address space of 80386.

- Virtual mode effective address i.e. segment offsets that exceed 64KB will cause an exception 13 as in Real mode.

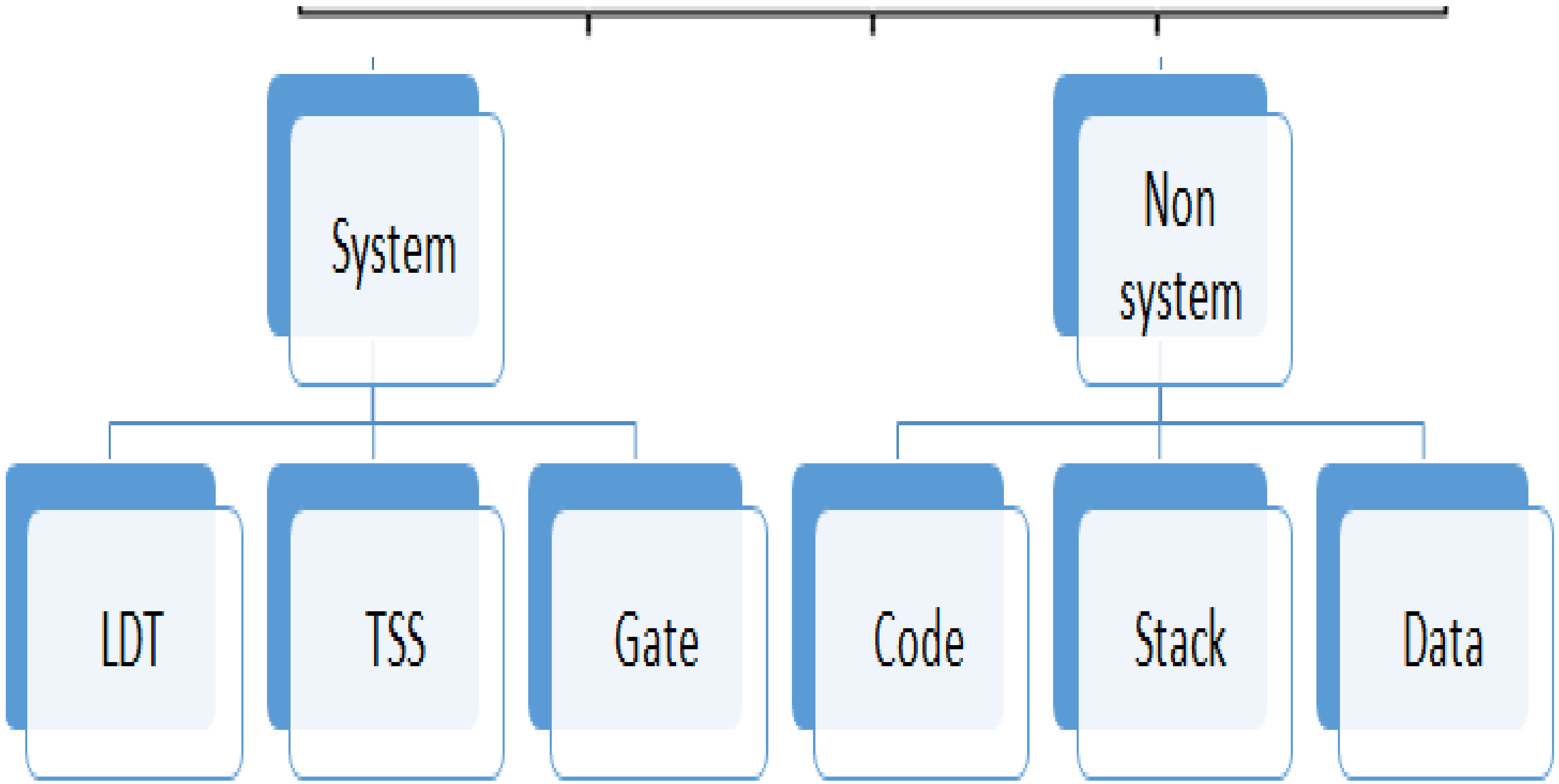
Memory Management

- The 80386 transforms **logical addresses** (i.e., addresses as viewed by programmers) **into physical address** (i.e., actual addresses in physical memory) **in two steps:**
  - **Segment translation**, in which a **logical address** (consisting of a segment selector and segment offset) are **converted to a linear address**.
  - **Page translation**, in which a **linear address** is converted to a **physical address**. This step is optional.

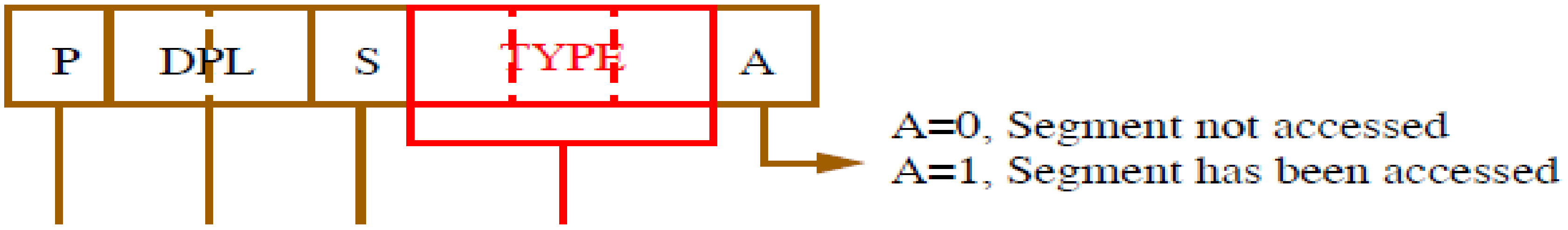
Segment Translation



Types of Segment Descriptor



Non System Descriptor



S=1

Non System

000	Data, read-only
001	Data, read/write
010	Stack, read-only
011	Stack, read/write
100	Code, execute-only
101	Code, execute/read
110	Code, execute-only, conforming
111	Code, execute/read, conforming

System Descriptor

- All system descriptors are present in GDT while some system descriptors are present in LDTs.
- Normally system segment descriptor are used by OS.
- The value of S in right access byte is 0.
- Their functions are fixed and specified by Intel.
- The type of system descriptor is indicated by type field
- The system segment descriptors have no Accessed bit, instead the type field (3 bits) is now extended to 4 bits.
- The system segment descriptors contain the information about tables (LDT), tasks(TSS) and gates (call gate, interrupt gate, task gate, trap gate) of the OS.

System Descriptor

S=0 System Descripors

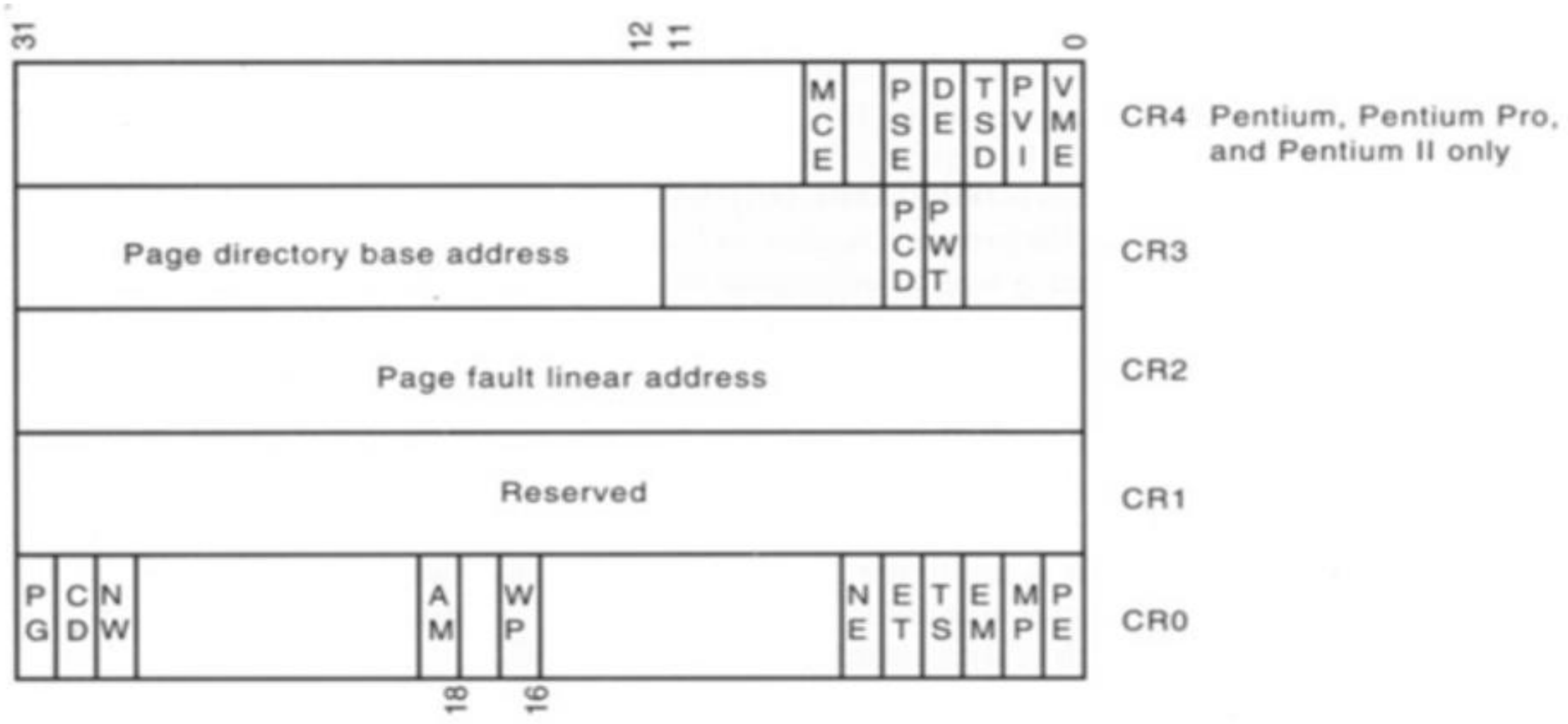
Type	Defines	Type	Defines
0	Reserved by Intel	8	Reserved by Intel
1	Available 80286 TSS (Task State Segment)	9	Available Intel 80286 TSS
2	LDT	A	Undefined
3	Busy 80286 TSS	B	Busy Intel 80386DX TSS
4	80286 Call Gate	C	Intel 80386DX Call gate
5	Task Gate	D	Undefined
6	80286 Interrupt Gate	E	80386DX Interrupt Gate
7	80286 Trap Gate	F	80386DX Trap Gate

Memory Paging

- The memory paging mechanism located within the 80386 allows any physical memory location to be assigned to any linear address.
- The linear address is defined as the address generated by a program.
- With the memory paging unit, the linear address is invisibly translated into any physical address, which allows an application written to function at a specific address to be re-located through the paging mechanism.

Paging Registers

- The paging unit is controlled by the contents of the microprocessor's control registers.



31
12
11
0
MCE
PSE
DE
TSD
PVI
VME
CR4 Pentium, Pentium Pro, and Pentium II only
Page directory base address
PCD
PWT
CR3
Page fault linear address
CR2
Reserved
CR1
PG
CD
NW
AM
WP
NE
ET
TS
EM
MP
PE
CR0
18
16

Paging Registers

- The registers important to the paging unit are CR0 and CR3.
- The leftmost bit (PG) position of CR0 selects paging when placed at a logic 1 level.
- If the PG bit is cleared (0), the linear address generated by the program becomes the physical address used to access memory.
- If the PG bit is set (1), the linear address is converted to a physical address through the paging mechanism.
- The paging mechanism functions in both the real and protected modes.

Paging Registers

- CR3 contains the page directory base address, and the PCD(Page Cache Disable) and PWT(page write through) bits. The PCD and PWT bits control the operation of the PCD and PWT pins on the microprocessor.
- If PCD is set (1), the PCD pin becomes a logic one during bus cycles that are not pages.
- This allows the external hardware to control the level 2 cache memory. (The level 2 cache memory is an external high-speed memory that functions as a buffer between the microprocessor and the main memory system.)

Paging Registers

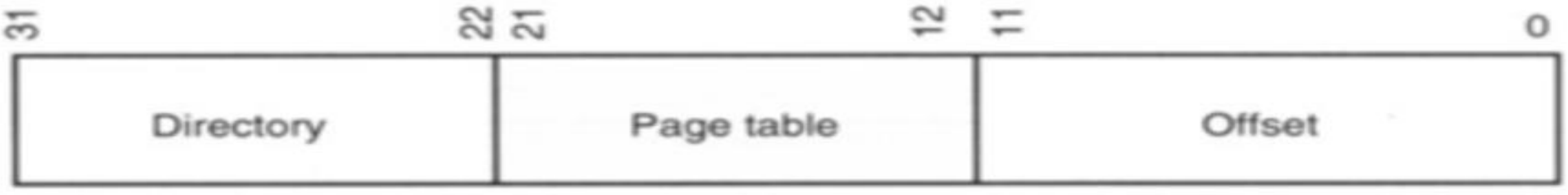
- The PWT bit also appears on the PWT pin, during bus cycles that are not pages, to control the write-through cache in the system.
- The page directory base address locates the page directory for the page translation unit.
- This address locates the page directory at any 4K boundary in the memory system because it is appended internally with a 000H.
- The page directory contains 1024 directory entries of 4 bytes each. Each page directory entry addresses a page table that contains 1024 entries.

Paging Registers

- The PWT bit also appears on the PWT pin, during bus cycles that are not pages, to control the write-through cache in the system.
- The page directory base address locates the page directory for the page translation unit.
- This address locates the page directory at any 4K boundary in the memory system.
- The page directory contains 1024 directory entries of 4 bytes each.
- Each page directory entry addresses a page table that contains 1024 entries.

Paging Registers

- The linear address, as it is generated by the software, is broken into three sections that are used to access the page directory entry, page table entry, and page offset address.
- Figure shows the linear address and its makeup for paging.
- The leftmost 10 bits address an entry in the page directory. For linear address 00000000H—003FFFFFH, the first entry of the page directory is accessed.
- Each page directory entry represents or repages a 4M-byte section of the memory system.
- The contents of the page directory select a page table that is indexed by the next 10 bits of the linear address (bit positions 12-21).



Paging Registers

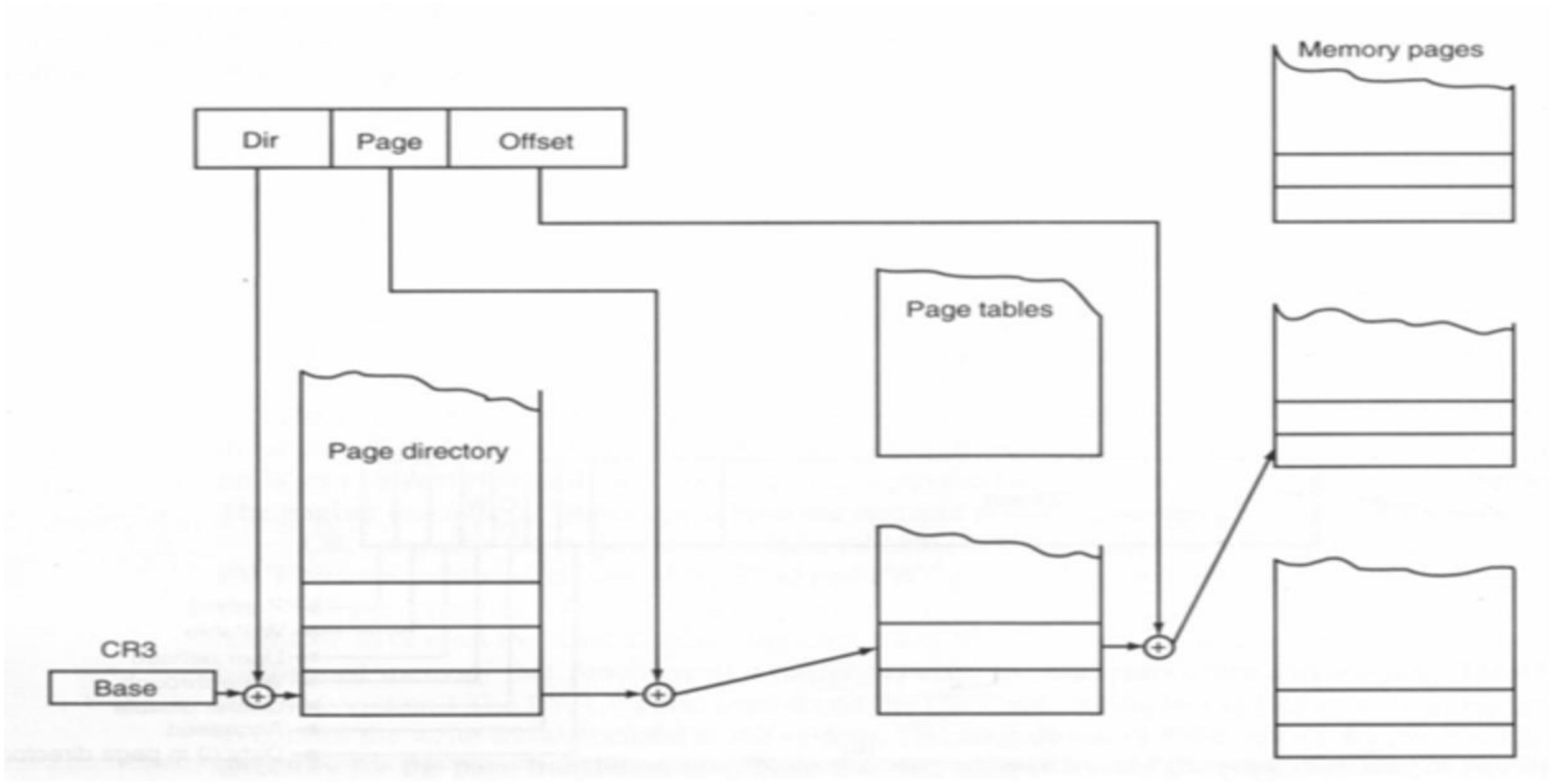
- This means that address 00000000H—00000FFFH selects page directory entry 0 and page table entry 0.
- This is a 4K-byte address range.
- The offset part of the linear address (bit positions 0-11) next selects a byte in the 4K-byte memory page.

Paging Registers

- Because the act of repaging a 4K-byte section of memory requires access to the page directory and a page table, which are both located in memory, Intel has incorporated a cache called the TLB (translation look-aside buffer).
- This means that the last 32 page table translations are stored in the TLB, so if the same area of memory is accessed, the address is already present in the TLB, and access to the page directory and page tables is not required.
- This speeds program execution.
- If a translation is not in the TLB, the page directory and page table must be accessed, which requires additional execution time.

The Page Directory and Page Table

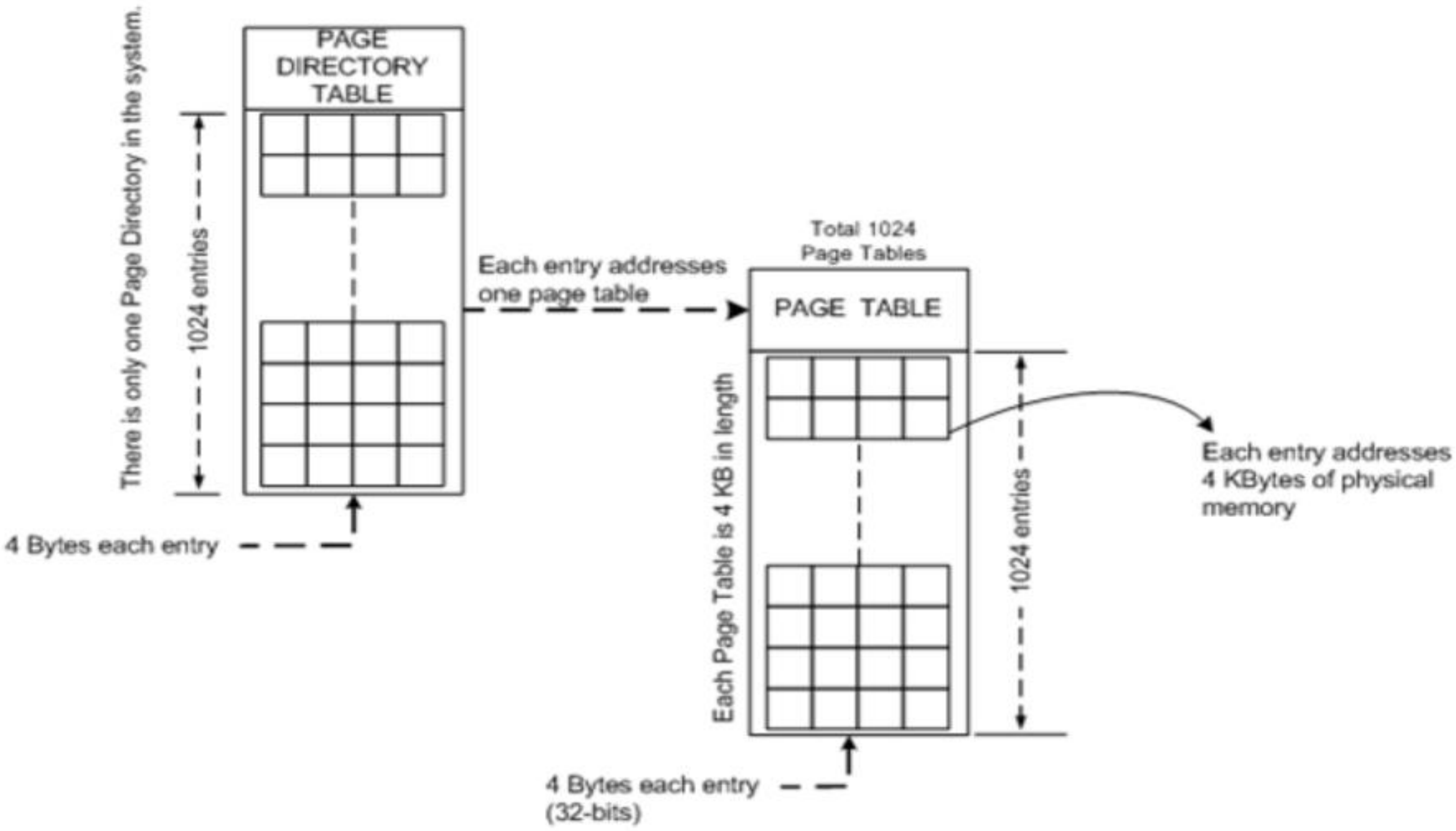
- Figure shows the page directory, a few page tables, and some memory pages.



The Page Directory and Page Table

- There is only one page directory in the system.
- The page directory contains 1024 doubleword addresses that locate up to 1024 page tables.
- The page directory and each page table are 4K bytes in length.
- If the entire 4G byte of memory is paged, the system must allocate 4K bytes of memory for the page directory, and 4K times 1024 or 4M bytes for the 1024 page tables.
- This represents a considerable investment in memory resources.

The Page Directory and Page Table


There is only one Page Directory in the system.
1024 entries
PAGE DIRECTORY TABLE
4 Bytes each entry
Each entry addresses one page table
Total 1024 Page Tables
PAGE TABLE
Each Page Table is 4 KB in length
1024 entries
Each entry addresses 4 KBytes of physical memory
4 Bytes each entry (32-bits)

The Page Directory and Page Table

- Here, the page directory contains four entries.
- Each entry in the page directory corresponds to 4M bytes of physical memory.
- The system also contains four page tables with 1024 entries each.
- Each entry in the page table repages 4K bytes of physical memory.
- This scheme requires a total of 16K of memory for the four page tables and 16 bytes of memory for the page directory.